การทำอักขระกราฟิกด้วย WORD PC

WORD PC
WORD PC

อาจหาญ สัตยารักษ์

การสร้างอักขระบนกราฟิกโหมดนั้นบ้างก็ว่าง่าย บ้างก็ว่ายาก แต่ทุกคนจะบอกเป็นเสียงเดียวกันว่า สิ่งที่ยากกว่าก็คือ การทำให้ได้ตัวอักขระสวย ๆ ต่างหาก แต่บทความนี้จะช่วยให้ผู้อ่านพบแนวทางใหม่ในการสร้างอักขระกราฟิกที่สะดวกอยู่หลายชั้นเพราะ

ประการที่หนึ่ง ไม่ต้องกำหนดรูปร่างของอักขระเอง หรือถ้าต้องการสร้างอักขระใหม่ก็ไม่ต้องเขียนโปรแกรมสร้างอักขระกันเลย เพราะอาศัยไฟล์ที่มีอยู่แล้วในราชวิถี WORD PC

ดังนั้นประการแรกจึงขอกราบขอบคุณ นพ.ชุษณะ มะกรสาร ชมรมไมโครคอมพิวเตอร์ โรงพยาบาลราชวิถี มา ณ ที่นี้เป็นอย่างสูงในฐานะเป็นผู้สร้างราชวิถี WORD PC มา และเรียนขออนุญาตใช้ไฟล์มาพร้อมนี้ด้วย

ประการที่สอง ต้องขอโทษผู้อ่านที่หัวข้อเรื่องอาจชวนให้ผู้อ่านเขว คิดว่าใช้ราชวิถี WORD PC จริง ๆ โดยขอเรียนว่า ผู้เขียนอาศัยราชวิถี WORD PC โดยการอ่านไฟล์ FONT1.NLQ, FONT2.NLQ มาสร้างอักขระกราฟิกขึ้นมาใช้เท่านั้น (ราชวิถี WORD PC ใช้สองไฟล์นี้เมื่อต้องการพิมพ์ข้อความลงเครื่องพิมพ์ โดย FONT1 จะเป็นตัวปกติ ส่วน FONT2 เป็นตัวเอียง) กับอาศัยไฟล์ FONTGEN.EXE เมื่อต้องการแก้ไขอักขระที่ต้องการใหม่

ศึกษาไฟล์ของราชวิถี WORD PC

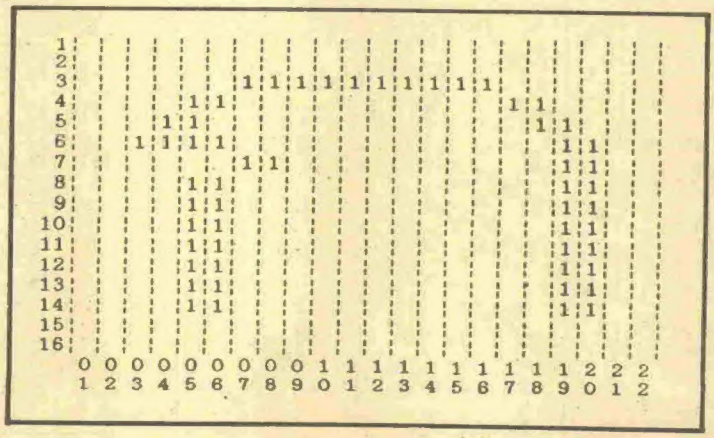
เมื่อเราใช้ไฟล์ FONTGEN.EXE ตรวจดูอักขระที่สร้างไว้จะพบว่าสร้างด้วยรูปขนาด 16 × 22 จุดทุกอักขระ ถ้าดูอักขระ ก จะเป็นดังรูปที่ 1 ซึ่งส่วนมากทิ้งด้านขอบว่างไว้ 2 คอลัมน์ (คอลัมน์ที่ 1,2,21 และ 22) เพื่อให้เป็นช่องว่างตอนพิมพ์ลงกระดาษและเราก็อาศัยมาเป็นช่องว่างของการแสดงอักขระกราฟิกด้วย

เราจะใช้คำว่า บิต เมื่อพูดถึงแถว คือ แถวที่ 1 ก็คือ บิตที่ 1 เป็นต้น เพราะรหัสที่เก็บจะเก็บในแนวตั้ง ที่บิต 1-16

อักขระภาษาไทยจะตั้งส่วนล่างสุดไว้ที่บิตที่ 14 ส่วนอักขระภาษาอังกฤษนั้นจะตั้งส่วนล่างสุดไว้ที่บิตที่ 12 เพราะว่าใช้บิตที่ 13-16 เขียนส่วนล่างของอักขระ g p q ฯลฯ ดังนั้นเวลาเรานำมาแสดงกราฟิกต้องกำหนดค่าตัวแปรเพื่อให้ระดับล่างของภาษาไทยและภาษาอังกฤษตรงกันด้วย (ผู้เขียนใช้ตัวแปรชื่อ Efont ในการนี้เมื่ออยู่ในโปรแกรมรูปที่ 9)

แต่การเก็บค่าจากรูปที่ 1 เข้าไปไว้ในไฟล์ FONT1.NLQ หรือ FONT2.NLQ นั้น อาศัยการเก็บเพื่อจะส่งรหัสไปควบคุมหัวเข็มของเครื่องพิมพ์ชนิด 9 เข็ม (ปกติทำงานเพียง 8 เข็ม แต่การพิมพ์แบบ NLQ จะใช้ครบทั้ง 9 เข็ม คือใช้เข็มที่ 2-9 ครั้งหนึ่ง และเข็มที่ 1-8 อีกครั้งหนึ่งโดยส่งรหัสครั้งละ 8 บิต ซึ่งเป็นผลให้ต้องพิมพ์สองครั้งต่อหนึ่งแถวตามที่ผู้ใช้ราชวิถี WORD PC ทราบนั่นเอง) โดยเก็บ 44 ไบต์ต่อหนึ่งตัวอักขระ เมื่อตรวจดูความยาวของไฟล์สองไฟล์นี้จะพบว่ามีความยาว 9856 ไบต์ เมื่อลองหารด้วย 44 จะได้ 224 ซึ่งบอกให้ทราบว่ามีเพียง 224 อักขระเท่านั้นที่ถูกเก็บไว้ในไฟล์ (เก็บในไฟล์เป็นข้อมูลลำดับที่ 0-223)

เนื่องจากต้องส่งรหัสให้เครื่อง



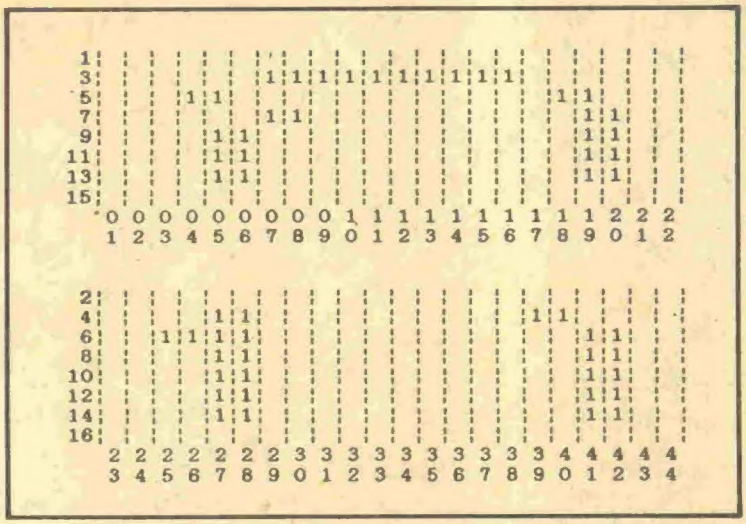
รูปที่ 1
ตัวอย่างอักขระ ก ที่สร้างด้วยไฟล์ FONTGEN.EXE ของราชวิถี WORD PC

พิมพ์สองชุดต่อหนึ่งอักขระ ราชวิถี WORD PC จึงเก็บรหัสไว้ดังนี้

ไบต์ที่ 1-22 เก็บรหัสเฉพาะบิตที่เป็นเลขคี่คือ 1,3,5,7,9,11,13,15 ของแต่ละคอลัมน์

ไบต์ที่ 23-44 เก็บรหัสเฉพาะบิตที่เป็นเลขคู่ คือ 2,4,6,8,10,12,14,16 ของแต่ละคอลัมน์

ดังนั้นอักขระ ก จากรูปที่ 1 เมื่อเก็บไฟล์จึงเป็นดังรูปที่ 2



รูปที่ 2 → การเก็บอักขระ ก จากรูปที่ 1 ลงไฟล์ FONT1.NLQ รูปบนคือ ไบต์ที่ 1-22 และรูปล่างคือ ไบต์ที่ 23-44

	0	1	2	3	4	5	6	7	8	9
0		!	"	#	$	%	&	'	(	)
10	*	+	,	-	.	/	0	1	2	3
20	4	5	6	7	8	9	:	;	<	=
30	>	?	@	A	B	C	D	E	F	G
40	H	I	J	K	L	M	N	O	P	Q
50	R	S	T	U	V	W	X	Y	Z	[
60	\	]	^	_	`	a	b	c	d	e
70	f	g	h	i	j	k	l	m	n	o
80	p	q	r	s	t	u	v	w	x	y
90	z	{	¦	}	~					
100	◌	◌	◌	◌	◌	◌	◌	◌		
110								◌	◌	◌
120	◌	◌	◌	◌	◌	◌	◌	◌		◌
130	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
140	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
150	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
160	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
170	◌	◌	◌		◌	◌	◌	◌	◌	◌
180	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
190	◌	◌	◌	◌	◌	◌	◌		◌	◌
200	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
210	◌	◌	◌	◌	◌	◌	◌	◌	◌	◌
220	◌	◌	◌							

รูปที่ 3 → ภาพแสดงลำดับของอักขระที่เก็บในไฟล์ FONT1.NLQ, FONT2.NLQ

เนื่องจากไม่มีการกำหนดรหัสของอักขระที่มีค่า แอสกี้ต่ำกว่า 32 ดังนั้นจึงเริ่มเก็บอักขระแรกไว้ที่ข้อมูลลำดับที่ 0 หรือเก็บอักขระไว้ที่ข้อมูลลำดับที่แอสกี้ของอักขระนั้น ลบด้วย 32 นั่นคือ

ข้อมูลลำดับที่ 1 คืออักขระที่มีแอสกี้เป็น 1 + 32 = 33 คืออักขระ "

ข้อมูลลำดับที่ 33 คืออักขระที่มีแอสกี้เป็น 33 + 32 = 65 คืออักขระ A

ข้อมูลลำดับที่ 129 คืออักขระที่มีแอสกี้เป็น 129 + 32 = 161 คืออักขระ ก

ข้อมูลลำดับที่ 170 คืออักขระที่มีแอสกี้เป็น 170 + 32 = 202 คืออักขระ อ

ข้อมูลลำดับที่ 222 คืออักขระที่มีแอสกี้เป็น 223 + 32 = 254 คืออักขระ ◌๎

ฯลฯ

รูปที่ 3 จะเป็นการแสดงข้อมูลที่ถูกเก็บลงในไฟล์จริง ๆ ขอให้สังเกตข้อมูลที่ 139-142 ที่อักขระยังไม่มีเชิงปรากฏในข้อมูลรวมทั้งข้อมูลที่ 162 ด้วย (ญ ฎ ฏ ฐ ฤ) ซึ่งถ้าพิมพ์หรือนำไปแสดงอักขระกราฟิกต้องเรียกเชิงมาแสดงร่วมด้วยเสมอ (อยู่ข้อมูลที่ 103-107)

ส่วน ไ ใ ๅ นั้นแม้จะดูข้อมูลที่ 178-181 มีอยู่ แต่เมื่อพิมพ์จริง ๆ ไม่ได้ใช้ส่วนนี้ แต่จะไปใช้ข้อมูลที่ 99-102 แทน คือ ข้อมูลที่ 99-100 เป็น ไ หรือข้อมูลที่ 99, 101 เป็น ใ

สังเกตสระผสมคือ ◌ั ◌ิ ◌ี ◌ึ ◌ื เหล่านี้จะถูกเก็บไว้ตั้งแต่ข้อมูลที่ 198 เป็นต้นไป ข้อมูลบางข้อมูลว่างอยู่เช่น ข้อมูลที่ 110-116 และข้อมูลที่ 117-128 ก็ถือว่าว่างได้เพราะการตีตารางในกราฟิกนั้น ใช้คำสั่งอื่นได้ง่ายกว่าอยู่แล้ว ดังนั้นเราอาจใช้ FONTGEN.EXE มาแก้ไขส่วนนี้ได้เช่น อาจใส่เลขไทย, สัญลักษณ์ทางวิทยาศาสตร์หรือคณิตศาสตร์ก็ได้

อย่าลืมว่าลำดับที่ของข้อมูลถ้าเปลี่ยนเป็นแอสกี้จะต้องบวกด้วย 32 เสมอ เวลาที่เราจะนำข้อมูลมาแสดงเป็นอักขระกราฟิกก็ต้องมาเขียนโปรแกรมตรวจสอบส่วนนี้ด้วยเช่นกัน ขณะเดียว

กันผู้อ่านคงมองเห็นแนวการนำลงพิมพ์ด้วยเครื่องพิมพ์เช่นเดียวกัน

เขียนโปรแกรมแสดงอักขระกราฟิก

ผู้เขียนขอเลือกเขียนด้วยปาสคาล (จะใช้เทอร์โบปาสคาล 4.0 หรือ 5.0 ก็ได้) ประการแรกก็กำหนดตัวแปรที่จะใช้งาน (รูปที่ 4) ตัวแปรดังนี้

bit กำหนดค่าประจำบิตเพื่อจะนำไปตรวจสอบการแสดงภาพด้วยคำสั่ง and ซึ่งค่าเป็น 0 หรือ 1 เช่น 64 and bit [6] จะมีค่าเป็น 1 ซึ่งหมายความว่าให้ลงจุด ณ บิตที่ 6, 3 and bit [0] จะมีค่า 1 ก็ลงจุด ณ บิตที่ 0, 3 and bit [1] ก็เท่ากับ 1 จึงลงจุด ณ บิตที่ 1 ด้วย

Font กำหนดไว้รับข้อมูล 44 ไบต์ของแต่ละอักขระ

Font1, Font2 กำหนดไว้รับอักขระอย่างละ 224 ข้อมูล

FontDisp ตัวแปรกำหนดอักขระที่ใช้ตรวจสอบการลงจุด

นอกนั้นก็เป็นตัวแปรใช้งานร่วมที่คิดว่าคงศึกษาได้

ชุดคำสั่ง OpenGraph ใช้เข้าสู่กราฟิกของจอโมโนโครม ใช้คู่กับไฟล์ HERC.BGI ในแผ่นดิสค์ด้วย

```
uses crt,graph;
const Bit :array[0..7] of byte=(1,2,4,8,16,32,64,128);
Type Font=array [1..44] of byte;
     FontRecord=RECORD
                  Fontbuff:Font;
                end;
     FileType  = file of FontRecord;
var  FontFile:Filetype;     k,FontCode,Mode:byte;
     Font1,Font2:array[0..223] of Font;
     FontDisp:Font;         ch:char;

PROCEDURE OpenGraph;
var
   GrDriver,GrMode :integer;
begin
   GrDriver :=7; GrMode:=0;
  InitGraph(GrDriver, GrMode, '');
end; { OpenGraph  }

PROCEDURE SaveFont;
var   FontRec:FontRecord;
 BEGIN
  clrscr; textcolor(0); TextBackground(7);
  writeln('                                                ');
  writeln(' Create Font for Graphics with WORD PC ');
  writeln('  By Arthan Sattayarak  Lampang 52000  ');
  writeln('                                                ');
  textcolor(3); TextBackground(0);  Gotoxy(1,7);
  writeln('Loading FONT1.NLQ');
Assign(FontFile,'font1.nlq');
Reset(FontFile);
   For k:=0 to 223 do
     begin
       Seek(FontFile,k);  Read(FontFile,FontRec);
       With FontRec Do Font1[k]:=FontBuff;
     end;
  Close(FontFile);
  writeln('Loading FONT2.NLQ');
Assign(FontFile,'font2.nlq');
Reset(FontFile);
   For k:=0 to 223 do
     begin
       Seek(FontFile,k);  Read(FontFile,FontRec);
       With FontRec Do Font2[k]:=FontBuff;
     end;
  Close(FontFile); FontCode:=0; Mode:=0;
end; { SaveFont }
```

รูปที่ 4 การกำหนดตัวแปร, ชุดคำสั่งเข้ากราฟิก, ชุดคำสั่งอ่านไฟล์มาเก็บในตัวแปรมิติ Font1, Font2

ชุดคำสั่ง SoveFont เพื่ออ่านข้อมูลจากไฟล์ FONT1.NLQ และ FONT2.NLQ มาเก็บไว้ในตัวแปรที่กำหนดไว้แล้ว สำหรับชุดคำสั่งนี้เริ่มด้วยการพิมพ์ข้อความแบบกลับขาวดำ แสดงชื่อโปรแกรมและชื่อผู้เขียนบอกให้รู้ว่ากำลังอ่านไฟล์ใด ซึ่งคิดว่าส่วนนี้ผู้อ่านคงทำความเข้าใจได้ หรือถ้าไม่เข้าใจก็อย่าไปสนใจ เพราะไม่เกี่ยวกับงานกราฟิกโดยตรง

รูปแบบที่ 1 การแสดงอักขระขนาด 16 × 16 จุด

แม้ว่าข้อมูลที่อ่านมาจะมีถึง 22 ไบต์ต่อการเก็บแบบ 8 บิตก็ตาม แต่เนื่องจากมักวางไว้หัวท้ายด้วย จึงเหลือส่วนที่เป็นอักขระปรากฏบนจอเพียง 16 ไบต์ ดังนั้นถ้านำมาแสดงเป็นภาพครบทุกจุดก็จะปรากฏอักขระขนาด 16 × 16 จุดเท่านั้น จึงเขียนส่วนการแสดงภาพนี้ว่าชุดคำสั่ง Dot16×16 ซึ่งมีรูปแบบคำสั่งดังนี้ (ขอให้ดูโปรแกรมในรูปที่ 5 ประกอบด้วย)

Dot16×16 (xx,yy:integer);

xx,yy คือจุดมุมล่างซ้ายที่จะให้อักขระนั้น ๆ ไปปรากฏ

Ypix เป็นตัวแปรมิติที่มีค่า 1,3,5,7,9,11,13,15 โดยผู้เขียนกำหนดจากข้างล่างขึ้นบน (คำสั่งที่เขียนว่า yy-Ypix [Dot] เพราะต้องการใช้แนวล่างของทุกอักขระอยู่ในแนวเดียวกัน แม้ขนาดของอักขระจะต่างกัน คือกำหนดให้ขนาดอื่น (จะกล่าวถึงต่อไป) แสดงในระดับเดียวกันเมื่อกำหนดค่า yy เดียวกันนั่นเอง

ส่วนนี้ผู้เขียนกำหนดตัวเลขกลับกันกับรูปที่ 2 ซึ่งกำหนดจาก 1-16 บนลงล่าง

Byte_No คือ ลำดับที่ในแต่ละอักขระมีค่า 1-44 ตามจำนวนไบต์ที่มีในหนึ่งอักขระ

OneByte คือ ค่าตัวเลขที่ถูกเก็บไว้ใน FontDisp[Byte_No] เพื่อนำมาตรวจสอบการลงจุดของอักขระนั้น ๆ

Check ตัวแปรกำหนดไว้เพื่อตรวจสอบการลงจุดว่าจะลงจุด ณ บิตที่เท่าใด (ทั้งหมด 16 บิต) โดยถ้า Check mod 2 = 0 คือเมื่อค่า Check เป็นเลขคู่ (0, 2,4...) จะตรวจการลงจุดของบิตที่เป็นเลขคี่ คือบิต 1,3,5...,15 นับจากล่างขึ้นไป ตัวอย่างเช่น เมื่อค่า Byte_No =1, Check = 0, Dot = 0 ดังนั้นเมื่อตรวจการลงจุดจากคำสั่ง OneByte and Bit[Dot]<>0 ก็จะลงจุดที่ yy-Ypix [Dot] + Check mod 2 นั้นคือลงที่บิตที่ Ypix[Dot] + Check mod 2 = 1-0 = 1 เป็นต้น

ถ้าค่า Check mod 2 = 1 ก็จะลงจุด ณ บิตที่เป็นเลขคู่คือ บิตที่ 0,2,4,..., 14 แทน

ค่าตัวแปร xx จะเปลี่ยนก็ต่อเมื่อค่า Byte_No mod 2 = 0 จากคำสั่ง xx : = xx +1 − Check mod 2; นั่นคือเปลี่ยนไปทุกครั้งที่ลงจุดครบหนึ่งคอลัมน์ (หนึ่งคอลัมน์ลงจุดซ้ำสองครั้ง)

คงอ่านโปรแกรมและเข้าใจแล้วว่า Byte_No ที่ 1-22 จะลงเฉพาะบิตที่เป็นเลขคี่ เมื่อ Byte_No เป็น 23-44 ก็จะลงจุด ณ บิตที่เป็นเลขคู่ เมื่อการ and เป็นจริง

ขอให้สังเกตการกำหนดค่า Byte_No ที่จะสลับไปมาคือจะทำงานด้วยค่า Byte_No ที่ 1,23,2,24,3,25,..., 22,44

รูปแบบที่ 2 การแสดงอักขระขนาด 8 × 16 จุด

คือการลงอักขระให้มีจุดในแนวนอน 8 จุด และแนวตั้ง 16 จุด ซึ่งแสดงว่าในแนวตั้งยังคงเหมือนกับในรูปแบบที่ 1 ดังนั้นการลงจุดเมื่อตรวจพบตามแนวตั้ง (บิตที่ 0-15) จึงเหมือนกับโปรแกรมในรูปที่ 5 การแสดงอักขระแบบนี้จึงเป็นดังโปรแกรมในรูปที่ 6 คือชุดคำสั่ง Dot8×16 มีรูปแบบดังนี้

Dot8×16(xx,yy: integer);

โดยค่าตัวแปรต่าง ๆ และหลักการทำงานก็คล้ายกัน

คำสั่งที่ต่างกันก็คือการกำหนดให้แสดงแนวนอนเพียง 11 คอลัมน์ (มีจุดปรากฏ 8 จุด) อยู่ที่การตัดคำสั่ง xx:= xx + 1 − Check mod 2; ออกไป แล้วนำคำสั่ง

if check mod 4 = 0 then xx:= xx + 1; มาแทน นั่นคือลงจุดซ้ำในคอลัมน์เดิม 4 ครั้งก่อนถึงขยับค่า xx ไปทางขวา 1 จุด ซึ่งมีผลทำให้ได้อักขระขนาด 8 × 16 จุดดังกล่าว

รูปแบบที่ 3 การแสดงอักขระขนาด 8 × 8 จุด

เนื่องจากคราวนี้เป็นการลงจุดที่มีขนาดเล็กที่สุดคือเป็น 8 × 8 จุด ดังนั้นในแนวนอนหรือคอลัมน์ก็คงต้องใช้แนวคิดเดียวกันกับรูปแบบที่ 2 ลงจุดเพียง 11 คอลัมน์เหมือนกัน แต่คราวนี้ผู้เขียนมาใช้วิธีเขียนอีกแบบหนึ่งกล่าวคือไม่ลงทุกไบต์ (ค่า Byte_No:=Byte_No +2; ซึ่งแสดงว่าข้ามการตรวจ Byte_No ที่เป็นเลขคี่ทั้งหมดไป หรือตรวจการลงจุดเฉพาะค่า Byte_No เป็น 2,4,6,...44(Byte_No เริ่มที่ค่าเป็น 2)

โปรแกรมจะเป็นดังรูปที่ 7 หรือชุดคำสั่ง Dot8×8(xx,yy:integer); โดยตัวแปรและความหมายของตัวแปรยังคงเดิม

การกำหนดค่า Ypix ใหม่ให้มีค่าเป็น 1,2,3,4,5,6,7,8 คือส่วนที่ทำให้ลงจุดได้ 8 บิต

การตรวจค่า Byte_No ก็เรียงลำดับไปเรื่อย ๆ โดยเพิ่มทีละ 2 ในช่วงแรก (Byte_No = 2-22) ก็เพิ่มค่า x ไปทีละ 1 เรื่อย แต่เมื่อค่า Byte_No =24 ก็กำหนดให้มาลงจุดที่คอลัมน์เริ่มต้น(x:=xx) ใหม่ซึ่งค่า yy จะถูกเพิ่มขึ้น

```
PROCEDURE Dot16x16(xx,Yy:integer);
const Ypix:array[0..7] of byte=(1,3,5,7,9,11,13,15);
var Byte_No,Dot:integer; OneByte,Check:byte;

begin
   Byte_No:=1; Check:=0;
 Repeat
  OneByte := FontDisp[Byte_No];
  for Dot := 0 to 7 do
    if OneByte and Bit[Dot]<>0 then
       putpixel(xx,yy-YPix[Dot]+Check mod 2,1);
    Check:=Check+1;  xx:=xx+1-Check mod 2;
  if Check mod 2=1 then Byte_No:=Byte_No+22;
  if check mod 2=0 then Byte_No:=Byte_No-21;
 until Byte_No>=44;
End;
```

รูปที่ 5 ชุดคำสั่งแสดงอักขระกราฟิกขนาด 16 × 16 จุด

```
PROCEDURE Dot8x16(xx,Yy:integer);
const Ypix:array[0..7] of byte=(1,3,5,7,9,11,13,15);
var Byte_No,Dot:integer; OneByte,Check:byte;

begin
   Byte_No:=1; Check:=0;
 Repeat
  OneByte := FontDisp[Byte_No];
  for Dot := 0 to 7 do
    if OneByte and Bit[Dot]<>0 then
              putpixel(xx,yy-YPix[Dot]+Check mod 2,1);
    Check:=Check+1;
  if Check mod 2=1 then Byte_No:=Byte_No+22;
  if check mod 2=0 then Byte_No:=Byte_No-21;
  if Check mod 4=0 then xx:=xx+1;
 until Byte_No>=43;
End;
```

รูปที่ 6 ชุดคำสั่งแสดงอักขระกราฟิกขนาด 8 × 16 จุด

อีกหนึ่ง (yy := yy + 1;) ดังนั้นในช่วงหลังนี้การลงจุดจะขยับลงมาข้างล่าง 1 จุด การลงจุดจริงจึงเป็นการลงในขอบเขต 9 บิตแนวตั้ง

รูปแบบที่ 4 การแสดงอักขระขนาด 16×8 จุด

รูปแบบสุดท้ายที่ผู้เขียนเสนอในครั้งนี้คือขนาด 16 × 8 จุด (ความจริงจะลง 9 บิตแนวตั้ง) ดังนั้นโปรแกรมเป็นดังรูปที่ 8 ซึ่งเป็นชุดคำสั่ง Dot 16×8 ถ้าผู้อ่านลองนำไปเปรียบเทียบกับโปรแกรมในรูปที่ 5 ชุดคำสั่ง Dot 16×16 จะพบว่าคล้ายกันมาก

ดังนั้นคำอธิบายจึงคงไม่ต้องมีก็ได้

ส่วนที่ทำให้ 2 ชุดคำสั่งนี้ต่างกันอยู่ที่การกำหนดว่า Ypix เท่านั้นเพราะคราวนี้เรากำหนดให้เป็นแค่ 8 บิตเท่านั้น Ypix เป็น 1,2,3,4,5,6,7,8 (เหมือนกับของ Dot8×8

การนำอักขระมาแสดงบนจอภาพ

เมื่อเราได้เขียนโปรแกรมแสดงอักขระ 4 แบบในแต่ละชุดคำสั่งไว้แล้ว ตอนนี้ก็มาถึงการนำเอาทั้งสี่ชุดคำสั่งมารวมกันเพื่อให้สะดวกในการแสดงข้อความที่ต้องการ

ชุดคำสั่ง PrintFont จึงเกิดขึ้นด้วยรูปแบบดังนี้

PrintFont (XStart, YStart: integer; Content: string);

ตัวแปรที่กำหนดมีดังนี้

XStart, YStart จุดที่กำหนดมาให้เริ่มแสดงข้อความ

Content คือข้อความที่ต้องการให้แสดงบนจอภาพ

ตัวแปรมิติ Xpix มีไว้ในการขยับจุดในแนวนอนไปทางขวามือ ตามขนาดของอักขระที่แสดง มีค่าเป็น 22,11,11, 22 ตามชุดคำสั่งในรูปแบบที่ 1-4

ตัวแปรมิติ Ypix กำหนดไว้สำหรับแสดง สระล่าง, สระบน, วรรณยุกต์ รวมถึงเชิงของอักขระบางตัว และการแสดง ไ ไ ไ ด้วย เพื่อให้การแสดงรูปอักขระเหล่านี้เป็นไปอย่างถูกต้อง

YDot ตัวแปรที่รองรับค่าตัวแปร Ypix ตามสถานะของการแสดงอักขระแต่ละแบบ

ASCII ตัวแปรเก็บค่ารหัสแอสกี้ของแต่ละตัวอักขระตามค่าจริง แต่ว่านำไปแสดงผลก็ลบออกด้วย 32 เพื่อให้ตรงกับรูปแบบของอักขระที่เก็บไว้ในตัวแปร Font1, Font2

Pass ตัวแปรกำหนดไว้เพื่อตรวจสอบในกรณีที่เป็นอักขระที่ต้องวาดรูปมากกว่าหนึ่งครั้ง เช่น ญ ฎ ฏ ฐ ไ ไ ้ เป็นต้น

Efont ตัวแปรกำหนดให้แสดงอักขระที่มีรหัสแอสกี้น้อยกว่า 127 ลงมา (ภาษาอังกฤษ) เพื่อให้ได้ระดับล่างตรงกับอักขระภาษาไทยในทุกขนาดของอักขระ ค่านี้จึงเป็น 0,1,2 ตามแต่กรณี

Mode ตัวแปรกำหนดว่าจะเลือกแสดงอักขระแบบใดจากรูปแบบที่ 1-4 โดยสามารถกำหนดไว้ในข้อความที่ส่งมา (Content) ดังนี้

1 แสดงอักขระขนาด 16 × 16
2 แสดงอักขระขนาด 8 × 16
3 แสดงอักขระขนาด 8 × 8
4 แสดงอักขระขนาด 16 × 8

โดยตรวจจากค่า Content[k] < # 5 ตามโปรแกรม

FontCode ตัวแปรกำหนดการเลือก font ที่แสดงว่าจะเป็น Font1 หรือ Font2 โดยกำหนดในข้อความได้เช่นกัน โดยมีค่าดังนี้

#1 คือเลือก font1 #12 เลือก font2 ตรวจจากค่า Content[k] แบบเดียวกัน

```
PROCEDURE Dot8x8(xx,yy:integer);
const Ypix:array[0..7] of byte=(1,2,3,4,5,6,7,8);
var Byte_No,Dot,x,y:integer; OneByte:byte;

begin
   Byte_No:=2; x:=xx; y:=yy;
 Repeat
  OneByte := FontDisp[Byte_No];
     for Dot := 0 to 7 do
      if OneByte and Bit[Dot]<>0 then
                  putpixel(x,Y-YPix[Dot],1);
   Byte_No:=Byte_No+2; x:=x+1; y:=yy;
   if Byte_no=24 then begin x:=xx; yy:=yy+1; y:=Yy end;
 until Byte_No>44;
End;
```

รูปที่ 7 ชุดคำสั่งแสดงอักขระกราฟิกขนาด 8 × 8 จุด

```
PROCEDURE Dot16x8(xx,yy:integer);
const Ypix:array[0..7] of byte=(1,2,3,4,5,6,7,8);
var Byte_No,Dot:integer; OneByte,check:byte;

begin
   Byte_No:=1; Check:=0;
 Repeat
  OneByte := FontDisp[Byte_No];
  for Dot := 0 to 7 do
    if OneByte and Bit[Dot]<>0 then
                putpixel(xx,yy-YPix[Dot]+check mod 2,1);
   Check:=Check+1; xx:=xx+1-Check mod 2;
  if Check mod 2=1 then Byte_No:=Byte_No+22;
  if Check mod 2=0 then Byte_No:=Byte_No-21;
 until Byte_No>=44;
End;
```

รูปที่ 8 ชุดคำสั่งแสดงอักขระกราฟิกขนาด 16 × 8 จุด

ค่า Mode, FontCode นี้กำหนดค่าเริ่มต้นไว้เป็น 0 ทั้งคู่ คือเป็น font1 ขนาด 16 x 16 (กำหนดค่านี้ไว้ในชุดคำสั่ง SaveFont บรรทัดท้าย ๆ

มาดูโปรแกรมการทำงานจะเป็นดังรูปที่ 9 ขออธิบายคร่าว ๆ ดังนี้

บวกข้อความ content ด้วยช่องว่าง 3 ตัว เพราะจะมีการตรวจสอบด้วยคำสั่ง Content [k + 2]

วนตรวจค่า Content[k] เพื่อตรวจคำสั่งการกำหนด fontCode, Mode ซึ่งอาจกำหนดไว้ติดกันได้จนกว่าจะมีค่า content[k] > #31

ถ้าค่าตรวจค่า content[k] พบว่าเป็นวรรณยุกต์ (คือได้ค่า content[k] > #233) และพบว่าค่า content[k – 1] = #207 (◌ำ) ก็ให้ข้ามไปเสีย (เพราะได้แสดงอักขระไปก่อนหน้าแล้ว ข้อความว่า ค่ำ, ซ้ำ ฯลฯ นี้ให้ป้อนดังนี้ ค◌ำ ◌่, ซ◌ำ ◌้ คือให้ป้อนสระ ◌ำ ก่อนวรรณยุกต์

กำหนดค่า ASCII, Ydot, Pass, Efont

ถ้าค่า content[k+1] > #223 ให้ตรวจสอบค่า content[k] ว่าเป็น #217 – #221 (สระ ◌ิ ถึง ◌ั) หรือไม่ ถ้าเป็นก็ให้เปลี่ยนรหัสแอสกี้ใหม่ ส่วนนี้คือการตรวจสระผสมเช่น ◌ิ◌้ นั่นเอง

ตรวจค่า content [k]> #214 คือตั้งแต่สระ ◌ุ เป็นต้นไป ถ้าใช่ก็ให้ลดค่า xx ลงตามขนาดของตัวแปรมิติ Xpix โดยใช้ค่า Mode มาเป็นตัวกำหนด และให้ค่าของตัวแปร YDot จากตัวแปรมิติ YPix เพื่อนำไปแสดงอักขระในตำแหน่งที่ถูกต้อง

ถ้าเป็นอักขระตั้งแต่สระ ◌ิ ขึ้นไป (ค่า content[k] > #216 ก็กำหนดค่า YDot ให้เป็นค่าลบเพื่อจะนำไปแสดงอักขระที่ด้านบนของอักขระอื่น

แต่ถ้าเป็นภาษาอังกฤษ content [k] < #127 ก็ให้ค่า Efont = 2 หากค่าตัวแปร Mode>1 ก็ให้ค่า Efont = 1 ส่วนนี้คือที่กล่าวมาแล้วเป็นการจัดให้ภาษาไทยและภาษาอังกฤษอยู่ในระดับเดียวกัน

ตรวจสอบว่าเป็น ไ ใ ไ หรือไม่ (content[k] มีค่า #210 – #212) ถ้าเป็นก็ให้ ASCII เป็น 131 เพื่อจะได้นำข้อมูลที่ 131-32 = 99 มาแสดงแทนก่อน

คราวนี้ก็จะมาถึงช่วงการตรวจสอบค่าต่าง ๆ จนกว่าตัวแปร Pass ไม่เท่ากับ 2 ขอให้ดูรูปอักขระที่เก็บไว้จริง (รูปที่ 3) ประกอบด้วย

กำหนดค่าของ FontDisp ตามค่าของ FontCode หมายถึงการเลือกตัวปกติหรือตัวเอียง

แสดงภาพของอักขระนั้น ๆ ตามค่าตัวแปร Mode (0,1,2 หรือ 3) ที่กำหนดมากับ Content เราจึงสามารถกำหนดให้แสดงอักขระได้หลาย ๆ แบบในบรรทัดเดียวกัน

ตรวจว่าเป็น ญ ฎ ฏ ฐ (#171 – #174) หรือเปล่า ถ้าเป็นก็กำหนดค่าแอสกี้ให้ใหม่โดยลดลง 36 ASCII: = ASCII – 36 และกำหนดค่า Pass = 2 เพื่อวนไปแสดงเชิงของอักขระแต่ละตัวให้ครบถ้วน

หากเป็นสระ โ ใ ไ (ซึ่งได้วาดรูป

```
PROCEDURE PrintFont(XStart,YStart:integer; content:string);
const Xpix:array[0..3] of byte=(22,11,11,22);
      YPix:array[0..3] of byte=(14,14,8,8);
var xx,Ydot:integer;  ASCII,Pass,Efont:byte;
begin
  xx:=XStart;  content:=content+'  ';
  for k:=1 to length(content)-2 do
    begin
REPEAT
 if (content[k]<#5) then Mode:=ord(content[k])-1;
 if (content[k]>#10) and (content[k]<#13) then
                    FontCode:=ord(content[k])-11;
 if (content[k]<#13) then  k:=k+1;
UNTIL content[k]>#31;
 if (content[k]>#223) and (content[k-1]=#207) then k:=k+1;
    ASCII:=ord(content[k]); Ydot:=0; Pass:=0; Efont:=0;
 if content[k+1]>#223 then
    begin
      CASE content[k] of
       #217 :begin ASCII:=(ord(content[k+1])+14); k:=k+1; end;
       #218 :begin ASCII:=(ord(content[k+1])+19); k:=k+1; end;
       #219 :begin ASCII:=(ord(content[k+1])+23); k:=k+1; end;
       #220 :begin ASCII:=(ord(content[k+1])+27); k:=k+1; end;
       #221 :begin ASCII:=(ord(content[k+1])+10); k:=k+1; end;
      end; { end case}
    end;
 if (content[k]>#214) then  begin xx:=xx-XPix[Mode];
       Ydot:=Ypix[Mode]; end;
 if (content[k]>#216) then Ydot:=-Ypix[Mode];
 if (content[k]<#127) then Efont:=2;
 if (Efont=2) and (Mode>1) then Efont:=1;
 if ((content[k]>#209) and (content[k]<#213)) then ASCII:=131;
REPEAT  Pass:=Pass+1;
      CASE FontCode of
         0: FontDisp:=Font1[ASCII-32];
         1: FontDisp:=Font2[ASCII-32];
      end;
      CASE Mode of
         0: Dot16x16(xx,YStart+Ydot+Efont);
         1: Dot8x16(xx,YStart+Ydot+Efont);
         2: Dot8x8(xx,YStart+Ydot+Efont);
         3: Dot16x8(xx,YStart+Ydot+Efont);
       end;
  if ((content[k]>#170) and (content[k]<#175)) and (Pass=1) then
    begin ASCII:=ASCII-36; Ydot:=Ypix[Mode]; Pass:=2; end;
  if ((content[k]>#209) and (content[k]<#213)) and (Pass=1) then
    begin ASCII:=ord(content[k])-78; Ydot:=-Ypix[Mode]; Pass:=2; end;
  if (content[k]=#194) and (Pass=1) then
    begin ASCII:=139; Ydot:=Ypix[Mode]; Pass:=2; end;
  if ((content[k+1]=#207) and (content[k+2]<#215)) and (Pass=1) then
    begin ASCII:=222; Ydot:=-Ypix[Mode]; Pass:=2; end;
  if ((content[k+1]=#207) and (content[k+2]>#223)) and (Pass=1) then
    begin ASCII:=ord(content[k+2])+6; Ydot:=-Ypix[Mode]; Pass:=2; end;
UNTIL (Pass=1) or (pass=3);
       xx:=xx+XPix[Mode];
 end;
end;
```

รูปที่ 9 ชุดคำสั่งที่นำข้อความที่กำหนดไปแสดงบนจอภาพ

ส่วนต้นทิ้งไว้แล้ว) ก็กำหนดส่วนต่อข้างบนเพิ่มให้วนวาดจนครบ

ตรวจตัว ฤ(content[k]= #194) ก็ให้วาดรูปที่ 139-32=107 ซึ่งคือส่วนท้ายของ ฤ เองเพิ่มเติม

ตรวจว่าตัวถัดไปเป็นสระ ำ (content[k+1] = #207) หรือไม่ ถ้าเป็นก็ตรวจว่าตัวถัดไปอีกสองตัว (content [k+2]>#223) เป็นวรรณยุกต์หรือไม่ ถ้าเป็นก็กำหนดการแสดงอักขระของวรรณยุกต์ก่อน จึงจะจบการวนรอบแสดงอักขระ

กำหนดค่า xx ให้เพิ่มไปทางขวาตามค่าตัวแปร Xpix

โปรแกรมหลักในการแสดงข้อความ

เราก็ได้รูปแบบของโปรแกรมมาจนครบแล้วคราวนี้ก็มาถึงการส่งข้อความไปทำงานจริง จะเป็นไปดังรูปที่ 10 โดยอย่าลืมว่าโปรแกรมที่สมบูรณ์พร้อมที่จะรันได้ต้องประกอบด้วย ชุดคำสั่งเรียงลำดับจากต้นไฟล์ไปท้ายไฟล์ ดังนี้

โปรแกรมในรูปที่ 4, 5, 6, 7, 8, 9, และ 10

เริ่มต้นด้วยการเรียกใช้ชุดคำสั่ง SaveFont ก่อน เพื่ออ่านข้อมูลมาเก็บไว้ตั้งแต่ต้น จากนั้นเข้าสู่กราฟิกโหมดด้วยคำสั่ง OpenGraph ส่วนที่แสดงข้อความก็ทำได้ดังบรรทัดถัดมา ซึ่งคิดว่าผู้อ่านคงทำความเข้าใจได้ไม่ยาก โดยเมื่อรันจะได้รูปแบบของอักขระของข้อความทั้งหมด ดังรูปที่ 11

```
{ ====================== MAIN ======================}
 begin
   SaveFont;   opengraph;
   Rectangle(0,0,719,347); Rectangle(2,2,717,345);
   PrintFont(60,35,#11+#1+'การทำอักขระกราฟิกด้วย WORD PC');
   PrintFont(13,70,#12+#1+'โดยอาศัยข้อมูลจากไฟล์ที่ใช้พิมพ์ลงกระดาษ');
   PrintFont(22,105,#11+#2+'คือไฟล์ FONT1.NLQ และ '+#12+
      'ไฟล์ FONT2.NLQ (ตัวเอียง)'+#11+#2+' ของราชวิถี WORD PC');
   PrintFont(73,140,#12+#2+'สามารถนำมาแสดงผลได้หลายรูปแบบ'+
      ' ดังที่ปรากฏข้อความในกรอบนี้');
   PrintFont(43,175,#11+#3+'นอกจากนี้ยังสามารถแก้ไข หรือ'+#2+
      'สร้าง Font ใหม่'+#3+'ตามที่ต้องการได้อีกมากมาย');
   PrintFont(43,205,#12+#3+'โดยใช้'+#2+'ไฟล์ FONTGEN.EXE ของราชวิถี'+#3+
      'ช่วยในการแก้ไขเพิ่มเติม');
   PrintFont(13,240,#11+#4+'สอบถามรายละเอียดอื่นๆได้ที่ ');
   PrintFont(38,270,#2+'อ.อาจหาญ สัตยารักษ์, โรงเรียนลำปางกัลยาณี'+
      ' อำเภอเมือง จังหวัดลำปาง 52000');
   PrintFont(13,300,#4+'จากความคิดนี้สามารถสร้างเป็นแบบอื่นๆ');
   PrintFont(23,330,#12+#2+'เช่น แบบขยาย, ตัวกลับขาวดำ, แสดงผลในแนวดิ่ง ฯลฯ'+
      #11+#3+' หรือใช้ไฟล์ FONTA?.NLQ');
  repeat ch:=readkey;  until ch>#31;
  closegraph;
 end
```

รูปที่ 10 โปรแกรมส่วนที่กำหนดข้อความที่จะแสดงบนจอภาพ

การทำอักขระกราฟิกด้วย WORD PC

โดยอาศัยข้อมูลจากไฟล์ที่ใช้พิมพ์ลงกระดาษ

คือไฟล์ FONT1.NLQ และ ไฟล์ *FONT2.NLQ (ตัวเอียง)* ของราชวิถี WORD PC

สามารถนำมาแสดงผลได้หลายรูปแบบ ดังที่ปรากฏข้อความในกรอบนี้

นอกจากนี้ก็ยังสามารถแก้ไข หรือสร้าง Font ใหม่ตามที่ต้องการได้อีกมากมาย

โดยใช้ไฟล์ FONTGEN.EXE ของราชวิถีช่วยในการแก้ไขเพิ่มเติม

สอบถามรายละเอียดอื่นๆได้ที่

อ.อาจหาญ สัตยารักษ์ โรงเรียนลำปางกัลยาณี อำเภอเมือง จังหวัดลำปาง 52000

จากความคิดนี้สามารถสร้างเป็นแบบอื่นๆ

เช่น แบบขยาย, ตัวกลับขาวดำ, แสดงผลในแนวดิ่ง ฯลฯ หรือใช้ไฟล์ FONTA?.NLQ

รูปที่ 11 ภาพที่ได้จากจอภาพเมื่อรันโปรแกรมให้แสดงอักขระแบบต่าง ๆ

นับว่าเราได้ใช้ประโยชน์จากไฟล์ของ WORD PC อีกทางหนึ่ง อย่าลืมว่า... เรายังสามารถกำหนดรูปแบบของอักขระได้อีกหลายแบบ เช่น แบบขยาย, แบบเป็นภาพที่มองจากกระจก, ภาพแบบหัวกลับ (คล้ายกับเป็นเงาในน้ำของข้อความด้านบน หรืออาจจะลองใช้ไฟล์อื่น ๆ เช่น Font1.nlq ซึ่งเก็บไว้อักขระละ 36 ไบต์ หรือ Font1.Lq1 ที่เก็บไว้ 108 ไบต์ดูก็ได้ ก็จะได้อักขระที่แสดงบนจอภาพได้อีกมากมาย

นอกจากนี้ก็อาจใช้ไฟล์ FONTGEN.EXE ของราชวิถี WORD PC แก้ไขอักขระเดิมได้ หวังว่าท่านคงสนุกกับการออกแบบอักขระกราฟิกมากยิ่งขึ้น